ESAME DI FONDAMENTI DI INFORMATICA T-2 del 16/07/2013
Proff. E. Denti – G. Zannoni
Tempo a disposizione: 4 ore MAX

NB: il candidato troverà nell'archivio ZIP scaricato da Esamix anche il software "Start Kit"

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)

La compagnia *Zann-O-Taxi*, operante nella ridente cittadina di *Zann-O-Town*, ha richiesto un'applicazione che simuli il funzionamento del tassametro dei propri mezzi, secondo la tipica tariffa mista a tempo e chilometri.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Il servizio taxi segue una tariffa progressiva a base multipla (tempo e chilometri), che si somma allo scatto iniziale, di importo differenziato in base all'orario (diurno o notturno) e alla giornata (feriale o festiva).

Lo scatto iniziale

La quota fissa iniziale dipende dall'orario:

- di giorno (dalle 6.00 alle 21.59): € 4.00
- di notte (dalle 22.00 alle 5.59): € 6.00

La tariffa "tempo e chilometri"

Mentre il taxi è in movimento, la tariffazione è applicata in base alla distanza percorsa; tuttavia, per non penalizzare il passeggero in presenza di code o traffico intenso, quando il mezzo è "quasi fermo" si applica una tariffazione a tempo. Entrambe le modalità di tariffazione operano a scatti (di valore diverso da tariffa a tariffa), nel seguente modo:

- per velocità ≤ 27 km/h: tariffa a tempo, uno scatto (da 15 € cent) ogni 12 s [tariffa T0]
- per velocità > 27 km/h: tariffe a distanza, a scaglioni progressivi

Per scaglioni progressivi si intende che il costo/km cambia man mano che il totale indicato dal tassametro (**escluso lo scatto iniziale**) aumenta. Più esattamente, a Zann-O-Town il costo/km aumenta quanto più il viaggio si allunga:

- per importi fino a € 10.00: uno scatto ogni 100 m (valore scatto: 25 € cent) [tariffa T1]
- sopra € 10.00 e fino a € 25.00: uno scatto ogni 85 m (valore scatto: 20 € cent) [tariffa T2]
- sopra € 25.00: uno scatto ogni 65 m (valore scatto: € 15 cent) [tariffa T3]

Algoritmo di funzionamento

Ogni secondo il tassametro riceve una rilevazione dello spazio percorso, che usa per calcolare la velocità media mantenuta a partire dall'ultimo scatto: in base a ciò decide se operare, in quell'instante, a tempo o a distanza, a seconda se la velocità misurata non supera/supera la soglia dei 27 km/h. Una volta stabilito se operare a tempo o a distanza, il tassametro verifica se dev'esserci uno scatto verificando rispettivamente una delle due condizioni:

$$\frac{spazio\ percorso}{distanza\ di\ scatto} \geq 1\ \ se\ opera\ a\ distanza, \qquad OPPURE \qquad \frac{tempo\ trascorso}{durata\ scatto} \geq 1\ se\ opera\ a\ tempo$$

A tal fine, il tassametro mantiene sia lo spazio totale percorso dall'ultimo scatto, sia il tempo totale trascorso dall'ultimo scatto, usando l'uno o l'altro secondo necessità. Se c'è uno scatto, questi totali sono entrambi azzerati e lo scatto è tariffato in base alla tariffa appropriata. Alla fine della corsa, eventuali metri/secondi residui, che non abbiano dato luogo a uno scatto, non sono tariffati.

Esempi

1. Corsa di soli 8 secondi alla velocità di 30 km/h (distanza totale percorsa: 58,33m) → tariffa a distanza T1, uno scatto ogni 100 m → 0 scatti; perciò il costo della corsa è pari alla sola quota fissa iniziale.
2. Corsa di soli 8 secondi alla velocità di 20 km/h (distanza totale percorsa: 44,44m) → tariffa a tempo T0, uno scatto ogni 12 s → 0 scatti; perciò il costo della corsa è pari alla sola quota fissa iniziale.
3. Corsa di 4 minuti alla velocità di 15 km/h (distanza totale percorsa: 1 km) → tariffazione a tempo T0, uno scatto ogni 12 s (= 5 scatti al minuto) → 20 scatti da 15 € cent l'uno → costo della corsa 3.00 € + quota iniziale
4. Corsa di 2 minuti alla velocità di 30 km/h (distanza totale percorsa: 1 km) → tariffazione a distanza T1, uno scatto ogni 100m → 10 scatti da 25 € cent l'uno → costo della corsa 2.50 € + quota iniziale
   [in generale, una corsa di 1km svolta a più di 27km/h costerà sempre 2,50 € + quota iniziale, a qualunque velocità]
5. Corsa di 2 min 30 sec a velocità differenziata, prima 1 min a 30 km/h (500m totali) poi 1 min 30 sec a 20 km/h (altri 500m) → prima parte a tariffa T1, 5 scatti da 25 cent = 1.25€; seconda parte a tariffa T0, 7 scatti da 15 cent = 1.05 € → costo della corsa 2.30 € + quota iniziale

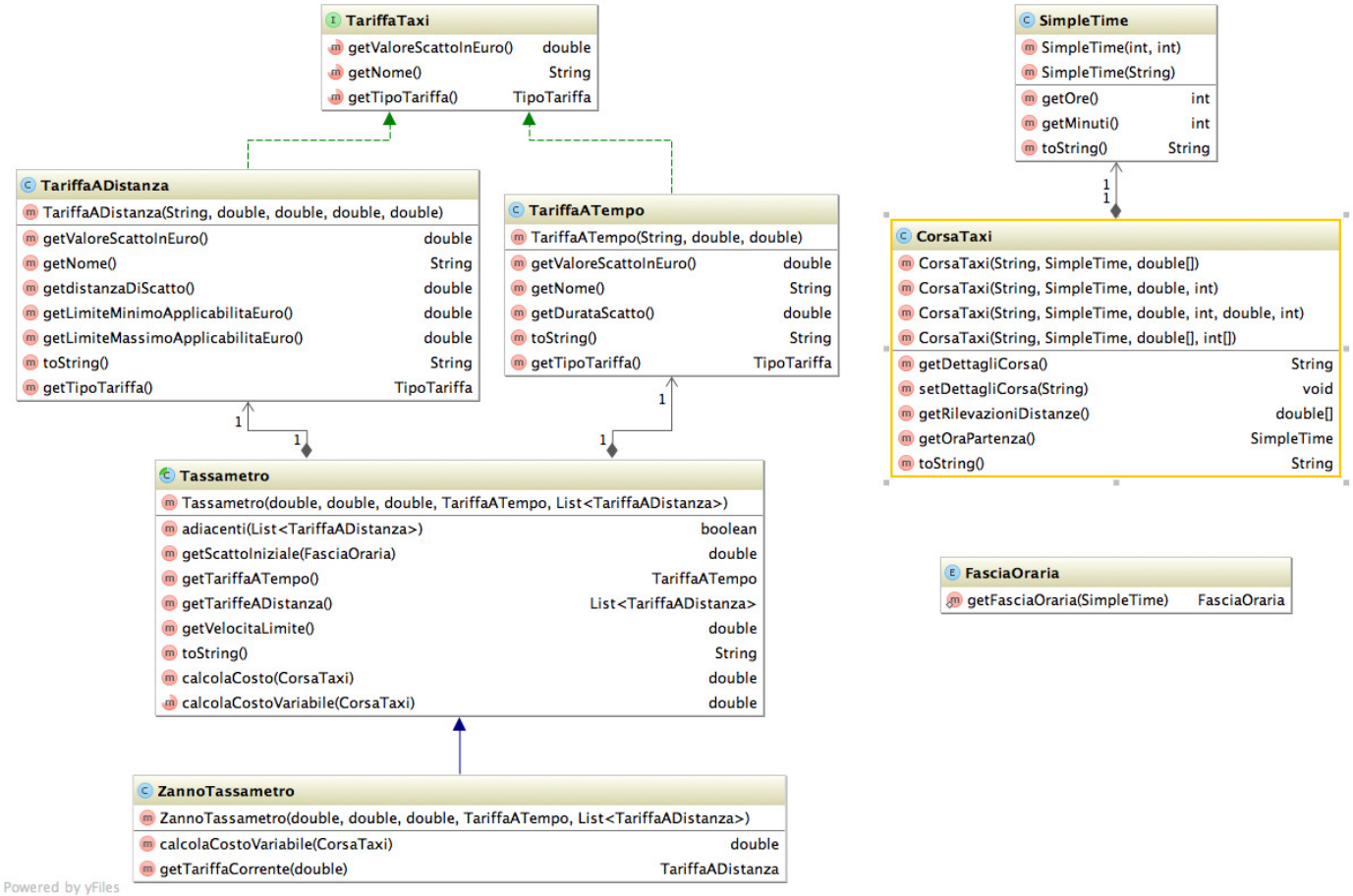
Il file binario CorseTaxi.bin contiene i dati di diverse corse (nel formato dettagliato più oltre): per ogni corsa è riportata una serie di *rilevazioni,* prese a un secondo una dall'altra, che indicano la distanza percorsa in metri.

Il file di testo Tariffe.txt contiene invece la specifica delle diverse tariffe, una per riga.

Parte 1 (punti: 20)

Dati (namespace zannotaxi.model) *(punti: 13)*

Il modello dei dati deve essere organizzato secondo il diagramma UML più sotto riportato.



SEMANTICA:

a) La classe ***SimpleTime*** (fornita nello start kit) rappresenta un semplice orario in termini di ore e minuti: fornisce gli opportuni accessor, comodi costruttori e un'adeguata ***toString.***

b) L'enumerativo ***FasciaOraria*** (fornito nello start kit) definisce i due valori ***DIURNA*** e ***NOTTURNA***; fornisce altresì un metodo statico ***getFasciaOraria(SimpleTime)*** che restituisce la giusta ***FasciaOraria*** in base all'orario passato.

c) La classe ***CorsaTaxi*** (fornita nello start kit) rappresenta una corsa di un taxi: essa è caratterizzata da un identificativo univoco, dall'orario di partenza e da una lista di *rilevazioni* (numeri reali), ciascuna delle quali specifica la distanza percorsa dall'inizio della corsa. Oltre agli opportuni metodi accessor e un'adeguata ***toString***, definisce un costruttore principale (l'unico realmente necessario) e due costruttori accessori di comodità (utili a fini di test).

d) L'interfaccia ***TariffaTaxi*** (fornita nello start kit) rappresenta la generica tariffa: dichiara i due metodi ***getValoreScatto*** che restituisce il valore di uno scatto in Euro e ***getNome*** che restituisce il nome della tariffa;

e) La classe ***TariffaATempo*** (fornita nello start kit) concretizza ***TariffaTaxi*** nel caso delle tariffe a tempo ed è caratterizzata, oltre che dal nome, dalla durata (in secondi) dello scatto. Tali valori sono forniti all'atto della costruzione e sono recuperabili tramite opportuni metodi accessor. Definisce anche un'adeguata ***toString***.

f) La classe ***TariffaADistanza*** (fornita nello start kit) concretizza ***TariffaTaxi*** nel caso delle tariffe a distanza ed è caratterizzata, oltre che dal nome, dalla distanza (in metri) che determina lo scatto, nonché da due valori (in Euro) che rappresentano l'intervallo di applicabilità della tariffa stessa. Tali valori sono forniti all'atto della costruzione e sono recuperabili tramite opportuni metodi accessor. Definisce un'adeguata ***toString***.

g) La classe astratta ***Tassametro*** (fornita nello start kit) rappresenta il generico tassametro, caratterizzato da:

- una *velocità limite*, in km/h, che rappresenta il confine fra tariffa a tempo e tariffa a distanza: più esattamente, oltre tale limite si applica la tariffa a distanza, mentre sotto tale limite si applica la tariffa a tempo; tale velocità è restituita dal metodo ***getVelocitaLimite***;
- un insieme di tariffe – di cui una sola a tempo e una o più a distanza – recuperabili tramite i metodi ***getTariffaATempo*** e ***getTariffeADistanza*** rispettivamente; poiché le tariffe a distanza possono essere più di una, ***getTariffeADistanza*** restituisce un array. Il costruttore di ***Tassametro*** verifica che i range di applicabilità delle tariffe a distanza fornite siano consecutivi e non determinino "buchi", lanciando ***IllegalArgumentException*** in caso contrario.

Il metodo ***calcolaCosto*** calcola il costo della ***CorsaTaxi*** data sommando la quota iniziale (che dipende solo dall'orario di inizio corsa) alla quota variabile: quest'ultima è calcolata dal metodo astratto ***calcolaCostoVariabile***.

h) **La classe *ZannoTassametro* (da realizzare)** concretizza ***Tassametro*** nel caso specifico di Zann-O-Taxi, implementando ***calcolaCostoVariabile*** secondo le specifiche dettagliate nel dominio del problema.
ATTENZIONE: si presti attenzione alle unità di misura (rilevazioni in metri, durate in secondi, velocità in km/h), ricordando che V[km/h] = V[m/s] * 3,6.

Nota: il significato dei parametri dei vari metodi è espresso dal loro nome (vedere start kit).

Persistenza (namespace zannotaxi.persistence) *(punti 7)*

Come già anticipato, il file binario CorseTaxi.bin contiene i dati di diverse corse: a tal fine, il primo dato è un intero che rappresenta il numero di corse taxi presenti nel file. Di seguito, per ogni corsa, sono presenti:

- due interi (ora e minuti dell'orario di inizio della corsa)
- una stringa che rappresenta la descrizione
- un array di double, che contiene le rilevazioni relative a quella corsa, ossia la distanza *in metri* percorsa dalla precedente rilevazione. (si ricordi che le rilevazioni avvengono a 1 secondo una dall'altra).

L' interfaccia ***CorseTaxiReader*** (fornita) dichiara il metodo ***getCorse*** che restituisce una lista di ***CorsaTaxi*** (preventivamente letta e memorizzata in fase di costruzione del reader stesso).

La classe *MyCorseTaxiReader* (da realizzare) implementa tale interfaccia: il costruttore provvede a leggere le ***CorsaTaxi*** dall'***InputStream*** fornito come argomento, lanciando ***BadFileFormatException*** (fornita) in caso sia di errore di formato sia di altri problemi di I/O (che devono essere intercettati e rilanciati esternamente come ***BadFileFormatException***). Il metodo ***getCorse*** restituisce la lista di ***CorsaTaxi*** preventivamente letta (e non lancia alcuna eccezione).

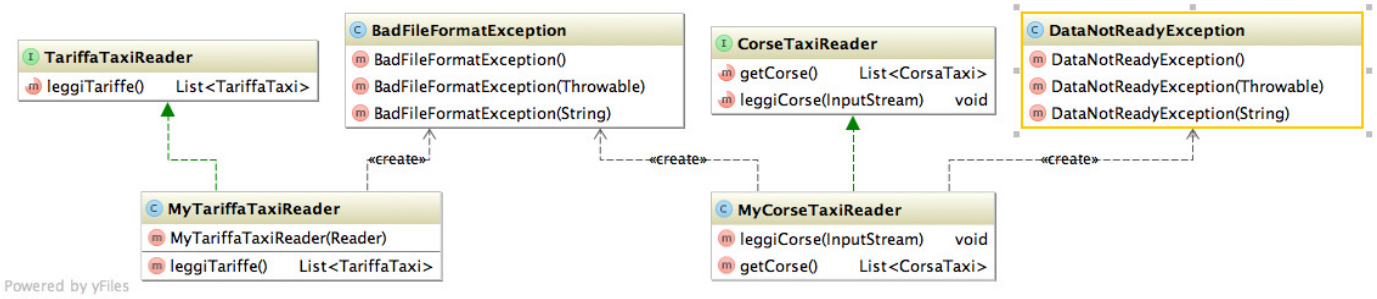
Il file di testo Tariffe.txt definisce una serie di tariffe, una per riga. Più esattamente, ogni riga contiene:

- un identificativo univoco della tariffa (senza spazi)
- la parola TEMPO o DISTANZA che ne qualifica la tipologia;
- nel caso delle tariffe a tempo, il tempo associato a uno scatto (con la relativa unità di misura) e il costo dello scatto stesso
- nel caso delle tariffe a distanza, la distanza associata a uno scatto (con la relativa unità di misura), il costo dello scatto stesso, seguito dalla specifica di un range di costi (determina l'applicabilità della tariffa), espressa come sequenza di due numeri (estremo inferiore, estremo superiore) o di un solo numero (estremo inferiore) nel caso di range superiormente illimitato; l'estremo inferiore si intende sempre escluso, quello

superiore sempre incluso. Nel caso di range superiormente illimitato, il limite superiore da impostare è il valore massimo assumibile dai valori double.

```
Formato del file Tariffe.txt
T0 TEMPO     12 s  0.15
T1 DISTANZA 100 m  0.25  0     10.00
T2 DISTANZA  85 m  0.35  10.00 25.00
T3 DISTANZA  65 m  0.50  25.00
```

L' interfaccia ***TariffaTaxiReader*** (fornita) dichiara il metodo ***leggiTariffe*** che legge da un ***Reader*** una lista di ***TariffaTaxi***.



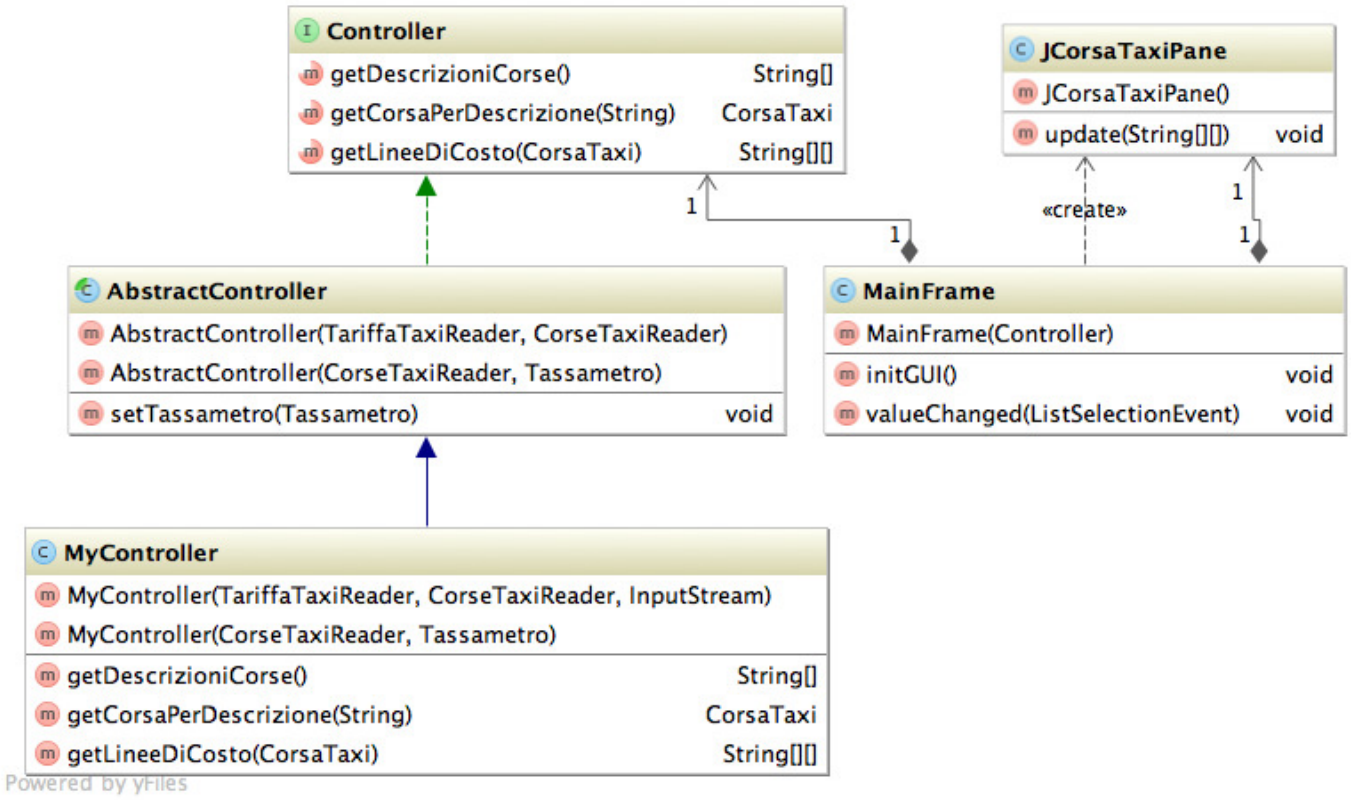
La classe *MyTariffaTaxiReader* (da realizzare) implementa tale interfaccia: il costruttore riceve il Reader da cui leggere, effettua la lettura facendo i necessari controlli e lanciando l'eccezione ***BadFileFormatException*** (fornita) in caso di errore di formato o di altri problemi di I/O, mentre il metodo ***leggiTariffe*** restituisce le tariffe lette.

Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.

Parte 2 (punti: 10)

Obiettivo dell'applicazione è simulare il funzionamento del tassametro con diverse corse. Pertanto, la GUI deve permettere di scegliere la corsa fra quelle disponibili per la simulazione, mostrando per ciascuna la fascia oraria dello scatto iniziale e il relativo importo, nonché l'importo della corsa calcolato secondo la tariffa mista tempo-distanza.

Controller (namespace zannotaxi.ui) *(punti 5)*



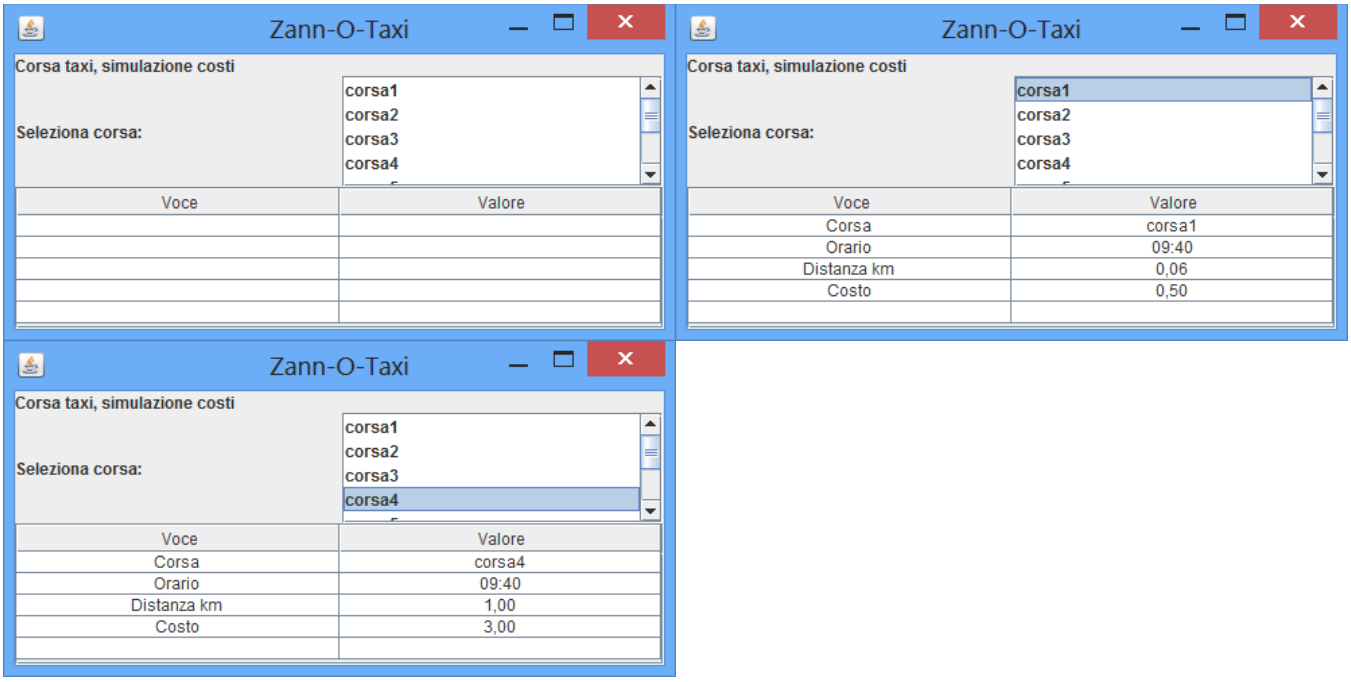
La classe *MyController* (da realizzare) implementa l'interfaccia ***Controller*** (fornita) derivando da ***AbstractController:*** tale classe contiene implementazioni parziali utili e definisce due costruttori. Pertanto, anche ***MyController*** deve definire due costruttori, richiamando il costruttore della classe base omologo. Il costruttore principale riceve in ingresso un ***TariffeReader***, un ***CorseTaxiReader*** e l'***InputStream*** da passare al metodo ***getCorse*** di ***CorseTaxiReader,*** mentre il costruttore accessorio (utile a fini di test) riceve in ingresso un ***CorseTaxiReader*** e un ***Tassametro.***

Il metodo ***getDescrizioniCorse*** restituisce l'elenco delle descrizioni delle corse taxi caricate, il metodo ***getCorsaPerDescrizione*** restituisce la ***CorsaTaxi*** corrispondente alla descrizione ricevuta come argomento, mentre il metodo ***getLineeTabella*** restituisce, data una ***CorsaTaxi***, una matrice di stringhe (ossia una sorta di tabella a due colonne di stringhe) in cui la prima colonna contiene una descrizione e l'altra il relativo valore: per comprenderne l'uso si guardi l'interfaccia utente e si osservi che questa matrice è utilizzata dal metodo ***update*** di ***JCorsaTaxiPane***.

È richiesto l'uso di un adeguato formattatore numerico (***NumberFormat***) per formattare la quarta riga della tabella (i costi) nel modo appropriato, ossia con due cifre decimali significative e opzionalmente il simbolo di valuta (€).

Interfaccia utente (namespace zannotaxi.ui) *(punti 5)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato in figura.



La classe ***Program*** (non mostrata nel diagramma UML, ma fornita nello start kit) contiene il main di partenza dell'intera applicazione.

La classe ***JCorsaTaxiPane*** (fornita nello start kit) mostra a video una tabella a due colonna di stringhe: il costruttore non richiede argomenti, mentre i valori da mostrare sono forniti in ingresso al metodo ***update*** come ***String[][]***.

La classe *MainFrame* (da realizzare) realizza la finestra principale, ed è così articolata:

- in alto, una JList, contenuta in apposito JScrollPane, mostra le corse taxi disponibili per la simulazione;
- quando l'utente ne sceglie una, nei campi di testo sottostanti vengono mostrati, tramite il ***JCorsaTaxiPane,*** i relativi dati.